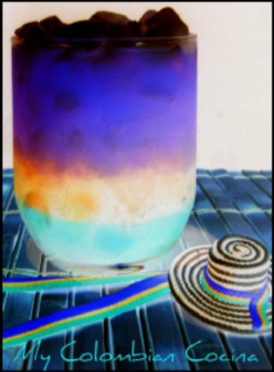
Data Structures I : O notation

Mauricio Toro
Deparment of Systems and Informatics
Universidad EAFIT

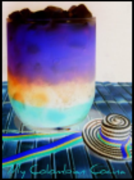
Cocktail of the day: Colombia



Disclaimer: Keep alcohol out of the hands of minors.

Cocktail of the day: Colombia

- 20 ml vodka
- 10 ml blue Curaçao
- 10 ml grenadine
- 10 ml lemon juice
- 60 ml orange juice



Logistic optimization uses trees

https://www.youtube.com/watch?v=rlC7hlORfxw

Complexity analysis

1. Number of instructions: $T(n)$
2. Asymptotic analysis: O notation
3. Rule of sums
4. Rule of products

`https://www.khanacademy.org/computing/computer-science/cryptography/modern-crypt/p/time-complexity-exploration`

Sum the elements of an array

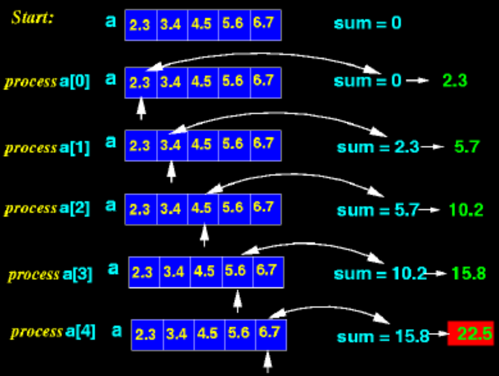


Figure: Array sum

Sum the elements of an array (2)

```
Proceso ArraySum
  Definir i, n, sum, A Como Entero;
  Leer n;
  sum <- 0;
  Dimension A[n];
  Para i <- 0 hasta n-1 con paso 1 Hacer
    sum <- sum + A[i];
  FinPara
  Escribir sum;
FinProceso
```

Sum the elements of an array (3)

```
Proceso ArraySum
  Definir i, n, sum, A Como Entero;
  Leer n;
  sum <- 0;
  Dimension A[n];
  Para i <- 0 hasta n-1 con paso 1 Hacer
    sum <- sum + A[i];
  FinPara
  Escribir sum;
FinProceso
```

Number of instructions $T(n) = ?$

Sum the elements of an array (4)

```
Proceso ArraySum
  Definir i, n, sum, A Como Entero;      // C1
  Leer n;                                // C2
  sum <- 0;                              // C3
  Dimension A[n];                        // C4
  Para i <- 0 hasta n-1 con paso 1 Hacer  // C5*n
    sum <- sum + A[i];                   // C6*n
  FinPara
  Escribir sum;                          // C7
FinProceso
```

$T(n) = c.n + c'$

Sum the elements of an array (5)

```
Proceso ArraySum
  Definir i, n, sum, A Como Entero;          // C1
  Leer n;                                    // C2
  sum <- 0;                                  // C3
  Dimension A[n];                            // C4
  Para i <- 0 hasta n-1 con paso 1 Hacer     // C5*n
    sum <- sum + A[i];                       // C6*n
  FinPara
  Escribir sum;                              // C7
FinProceso
```

$T(n) = c.n + c'$ is $O(n)$... why?

O notation

- Big O notation describes the limiting behavior of a function when the argument tends towards a particular value or infinity.
- Big O notation is used to classify algorithms by how they respond to changes in input size.

O notation

- Big O notation describes the limiting behavior of a function when the argument tends towards a particular value or infinity.
- Big O notation is used to classify algorithms by how they respond to changes in input size.

O notation (2)

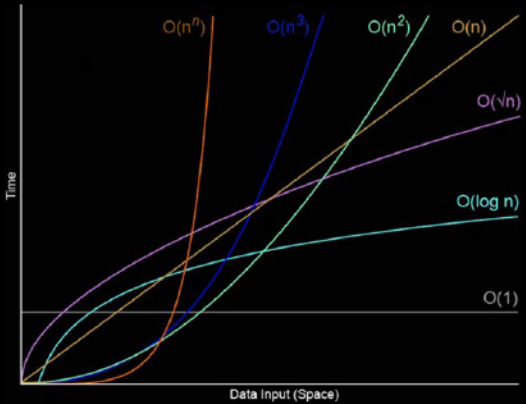


Figure: O notation plots

O cheat sheet

http://http://bigocheatsheet.com/

O notation (3)

- Let T and f two functions. One writes
  $T(n) = O(f(n))$ as $n \to \infty$
- if and only if, there is a constant M and a number n_0 such that
  $|T(n)| \leq M|f(n)|$ for all $n \geq n_0$
- In Computer Science, we say
  $T(n)$ is $O(f(n))$.

Sum the elements of an array (6)

```
Definir i, n, sum, A Como Entero;      // C1
Leer n;                                // C2
sum <- 0;                              // C3
Dimension A[n];                        // C4
Para i <- 0 hasta n-1 con paso 1 Hacer  // C5*n
  sum <- sum + A[i];                   // C6*n
FinPara
Escribir sum;                          // C7
```

$T(n) = c.n + c'$ is $O(n)$... why?

Sum the elements of an array (6)

```
Definir i, n, sum, A Como Entero;      // C1
Leer n;                                // C2
sum <- 0;                              // C3
Dimension A[n];                        // C4
Para i <- 0 hasta n-1 con paso 1 Hacer  // C5*n
  sum <- sum + A[i];                   // C6*n
FinPara
Escribir sum;                          // C7
```

$T(n) = c.n + c'$ is $O(n)$... why?

Rule of sums

- f_1 is $O(g_1)$ and f_2 is $O(g_2) \Rightarrow f_1 + f_2$ is $O(g_1 + g_2)$
- Corollary $O(f + g) = O(f) + O(g)$
- Corollary $O(f + g) = O(max(f, g))$
- Example $O(c.n + c') = O(c.n) + O(c')$
- Example $O(c.n + c') = O(n)$

Rule of sums

- f_1 is $O(g_1)$ and f_2 is $O(g_2) \Rightarrow f_1 + f_2$ is $O(g_1 + g_2)$
- Corollary $O(f+g) = O(f) + O(g)$
- Corollary $O(f+g) = O(max(f,g))$
- Example $O(c.n + c') = O(c.n) + O(c')$
- Example $O(c.n + c') = O(n)$

Rule of sums

- f_1 is $O(g_1)$ and f_2 is $O(g_2) \Rightarrow f_1 + f_2$ is $O(g_1 + g_2)$
- Corollary $O(f + g) = O(f) + O(g)$
- Corollary $O(f + g) = O(max(f, g))$
- Example $O(c.n + c') = O(c.n) + O(c')$
- Example $O(c.n + c') = O(n)$

Rule of sums

- f_1 is $O(g_1)$ and f_2 is $O(g_2) \Rightarrow f_1 + f_2$ is $O(g_1 + g_2)$
- Corollary $O(f+g) = O(f) + O(g)$
- Corollary $O(f+g) = O(max(f,g))$
- Example $O(c.n + c') = O(c.n) + O(c')$
- Example $O(c.n + c') = O(n)$

Rule of sums

- f_1 is $O(g_1)$ and f_2 is $O(g_2) \Rightarrow f_1 + f_2$ is $O(g_1 + g_2)$
- Corollary $O(f + g) = O(f) + O(g)$
- Corollary $O(f + g) = O(max(f, g))$
- Example $O(c.n + c') = O(c.n) + O(c')$
- Example $O(c.n + c') = O(n)$

How do we compute $max(f, g)$?

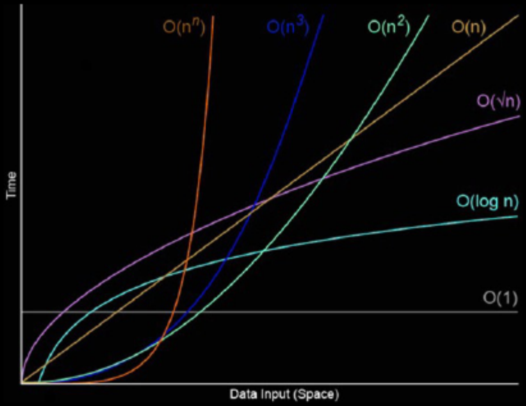


Figure: O notation plots

Rule of Products

- f_1 is $O(g_1)$ and f_2 is $O(g_2) \Rightarrow f_1 f_2$ is $O(f_1 f_2)$
- Corollary $O(fg) = O(f)O(g)$
- Corollary $O(c.g) = c.O(g) = O(g)$
- Example $O(c.n) = (n)$
- Example $O(c.n) = C.O(n) = O(n)$

Rule of Products

- f_1 is $O(g_1)$ and f_2 is $O(g_2) \Rightarrow f_1 f_2$ is $O(f_1 f_2)$
- Corollary $O(fg) = O(f)O(g)$
- Corollary $O(c.g) = c.O(g) = O(g)$
- Example $O(c.n) = (n)$
- Example $O(c.n) = C.O(n) = O(n)$

Rule of Products

- f_1 is $O(g_1)$ and f_2 is $O(g_2) \Rightarrow f_1 f_2$ is $O(f_1 f_2)$
- Corollary $O(fg) = O(f)O(g)$
- Corollary $O(c.g) = c.O(g) = O(g)$
- Example $O(c.n) = (n)$
- Example $O(c.n) = C.O(n) = O(n)$

Rule of Products

- f_1 is $O(g_1)$ and f_2 is $O(g_2) \Rightarrow f_1 f_2$ is $O(f_1 f_2)$
- Corollary $O(fg) = O(f)O(g)$
- Corollary $O(c.g) = c.O(g) = O(g)$
- Example $O(c.n) = (n)$
- Example $O(c.n) = C.O(n) = O(n)$

Rule of Products

- f_1 is $O(g_1)$ and f_2 is $O(g_2) \Rightarrow f_1 f_2$ is $O(f_1 f_2)$
- Corollary $O(fg) = O(f)O(g)$
- Corollary $O(c.g) = c.O(g) = O(g)$
- Example $O(c.n) = (n)$
- Example $O(c.n) = C.O(n) = O(n)$

Sum the elements of an array: Proof

1. $T(n) = c.n + c'$
2. $c.n + c'$ is $O(c.n + c')$, by Definition of O
3. $O(c.n + c') = O(c.n)$, by Rule of Sums
4. $O(c.n) = O(n)$, by Rule of Products
5. Therefore, $T(n) = c.n + c'$ is $O(n)$.

Sum the elements of an array: Proof

1. $T(n) = c.n + c'$
2. $c.n + c'$ is $O(c.n + c')$, by Definition of O
3. $O(c.n + c') = O(c.n)$, by Rule of Sums
4. $O(c.n) = O(n)$, by Rule of Products
5. Therefore, $T(n) = c.n + c'$ is $O(n)$.

Sum the elements of an array: Proof

1. $T(n) = c.n + c'$
2. $c.n + c'$ is $O(c.n + c')$, by Definition of O
3. $O(c.n + c') = O(c.n)$, by Rule of Sums
4. $O(c.n) = O(n)$, by Rule of Products
5. Therefore, $T(n) = c.n + c'$ is $O(n)$.

Sum the elements of an array: Proof

1. $T(n) = c.n + c'$
2. $c.n + c'$ is $O(c.n + c')$, by Definition of O
3. $O(c.n + c') = O(c.n)$, by Rule of Sums
4. $O(c.n) = O(n)$, by Rule of Products
5. Therefore, $T(n) = c.n + c'$ is $O(n)$.

Sum the elements of an array: Proof

1. $T(n) = c.n + c'$
2. $c.n + c'$ is $O(c.n + c')$, by Definition of O
3. $O(c.n + c') = O(c.n)$, by Rule of Sums
4. $O(c.n) = O(n)$, by Rule of Products
5. Therefore, $T(n) = c.n + c'$ is $O(n)$.

Max element of an array

```
Proceso MaxElement
  max <- A[0];
  Para i <- 1 hasta n-1 con paso 1 Hacer
    Si A[i] > max Entonces
      max <- A[i];
    FinSi
  FinPara
  Escribir sum;
FinProceso
```

$T(n) = ?$

Max element of an array (2)

```
Proceso MaxElement
  max <- A[0];                                   // c1
  Para i <- 1 hasta n-1 con paso 1 Hacer  // c2*n + c3
    Si A[i] > max Entonces                       // c4*n
      max <- A[i];                              // c5*n
    FinSi
  FinPara
  Escribir sum;                                  // c6
FinProceso
```

$T(n) = c.n + c'$

Max element of an array (3)

```
Proceso MaxElement
  max <- A[0];                                   // c1
  Para i <- 1 hasta n-1 con paso 1 Hacer  // c2*n + c3
    Si A[i] > max Entonces                       // c4*n
      max <-  A[i];                            // c5*n
    FinSi
  FinPara
  Escribir sum;                                  // c6
FinProceso
```

$T(n) = c.n + c'$ is $O(n)$... why?

Max element of an array: Proof

1. $T(n) = c.n + c'$
2. $c.n + c'$ is $O(c.n + c')$, by Definition of O
3. $O(c.n + c') = O(c.n)$, by Rule of Sums
4. $O(c.n) = O(n)$, by Rule of Products
5. Therefore, $T(n) = c.n + c'$ is $O(n)$.

Max element of an array: Proof

1. $T(n) = c.n + c'$
2. $c.n + c'$ is $O(c.n + c')$, by Definition of O
3. $O(c.n + c') = O(c.n)$, by Rule of Sums
4. $O(c.n) = O(n)$, by Rule of Products
5. Therefore, $T(n) = c.n + c'$ is $O(n)$.

Max element of an array: Proof

1. $T(n) = c.n + c'$
2. $c.n + c'$ is $O(c.n + c')$, by Definition of O
3. $O(c.n + c') = O(c.n)$, by Rule of Sums
4. $O(c.n) = O(n)$, by Rule of Products
5. Therefore, $T(n) = c.n + c'$ is $O(n)$.

Max element of an array: Proof

1. $T(n) = c.n + c'$
2. $c.n + c'$ is $O(c.n + c')$, by Definition of O
3. $O(c.n + c') = O(c.n)$, by Rule of Sums
4. $O(c.n) = O(n)$, by Rule of Products
5. Therefore, $T(n) = c.n + c'$ is $O(n)$.

Max element of an array: Proof

1. $T(n) = c.n + c'$
2. $c.n + c'$ is $O(c.n + c')$, by Definition of O
3. $O(c.n + c') = O(c.n)$, by Rule of Sums
4. $O(c.n) = O(n)$, by Rule of Products
5. Therefore, $T(n) = c.n + c'$ is $O(n)$.

Multiplication tables

```
for (int i = 1; i <= n; i++)
     for (int j = 1; j <= n; j++)
          print(i+"*"+j+"="+i*j);
```

$T(n) = ?$

Multiplication tables (2)

```
for (int i = 1; i <= n; i++) // C1*n
    for (int j = 1; j <= n; j++) //C2*n^2
        print(i+"*"+j+"="+i*j); //C3*n^2
```

$T(n) = (c_2 + c_3)n^2 + c_1.n$ is $O(n^2)$

Multiplication tables: Proof

1. $T(n) = (c_2 + c_3)n^2 + c_1 n$
2. $(c_2 + c_3)n^2 + c_1 n$ is $O((c_2 + c_3)n^2 + c_1 n)$, by Definition of O
3. $O((c_2 + c_3)n^2 + c_1 n) = O((c_2 + c_3)n^2)$, by Rule of Sums
4. $O((c_2 + c_3)n^2) = O(n^2)$, by Rule of Products
5. Therefore, $T(n) = (c_2 + c_3)n^2 + c_1 n$ is $O(n^2)$.

Multiplication tables: Proof

1. $T(n) = (c_2 + c_3)n^2 + c_1 n$
2. $(c_2 + c_3)n^2 + c_1 n$ is $O((c_2 + c_3)n^2 + c_1 n)$, by Definition of O
3. $O((c_2 + c_3)n^2 + c_1 n) = O((c_2 + c_3)n^2)$, by Rule of Sums
4. $O((c_2 + c_3)n^2) = O(n^2)$, by Rule of Products
5. Therefore, $T(n) = (c_2 + c_3)n^2 + c_1 n$ is $O(n^2)$.

Multiplication tables: Proof

1. $T(n) = (c_2 + c_3)n^2 + c_1 n$
2. $(c_2 + c_3)n^2 + c_1 n$ is $O((c_2 + c_3)n^2 + c_1 n)$, by Definition of O
3. $O((c_2 + c_3)n^2 + c_1 n) = O((c_2 + c_3)n^2)$, by Rule of Sums
4. $O((c_2 + c_3)n^2) = O(n^2)$, by Rule of Products
5. Therefore, $T(n) = (c_2 + c_3)n^2 + c_1 n$ is $O(n^2)$.

Multiplication tables: Proof

1. $T(n) = (c_2 + c_3)n^2 + c_1 n$
2. $(c_2 + c_3)n^2 + c_1 n$ is $O((c_2 + c_3)n^2 + c_1 n)$, by Definition of O
3. $O((c_2 + c_3)n^2 + c_1 n) = O((c_2 + c_3)n^2)$, by Rule of Sums
4. $O((c_2 + c_3)n^2) = O(n^2)$, by Rule of Products
5. Therefore, $T(n) = (c_2 + c_3)n^2 + c_1 n$ is $O(n^2)$.

Multiplication tables: Proof

1. $T(n) = (c_2 + c_3)n^2 + c_1 n$
2. $(c_2 + c_3)n^2 + c_1 n$ is $O((c_2 + c_3)n^2 + c_1 n)$, by Definition of O
3. $O((c_2 + c_3)n^2 + c_1 n) = O((c_2 + c_3)n^2)$, by Rule of Sums
4. $O((c_2 + c_3)n^2) = O(n^2)$, by Rule of Products
5. Therefore, $T(n) = (c_2 + c_3)n^2 + c_1 n$ is $O(n^2)$.

Insertion sort

Figure: Insertion sort

Insertion sort

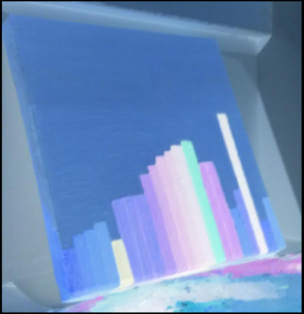


Figure: Mauricio Toro, “Insertion Sort”. In collection “Neoplasticistic sorting algorithms” (2015).

Insertion sort

https://www.youtube.com/watch?v=8oJS1BMKE64

Insertion sort (2)

```
Proceso InsertionSort
  Para i <- 0 hasta n-1 Hacer
    j <- i;
    Mientras j > 0 && A[j-1] > A[j] Hacer
      temp <- A[j];
      A[j] <- A[j-1];
      A[j-1] <- temp;
      j <- j - 1;
    FinMientras
  FinPara
FinProceso
```

$T(n) = ?$

Insertion sort (2)

```
Proceso InsertionSort
  Para i <- 0 hasta n-1 Hacer // c1*n + c2
    j <- i; // n
    Mientras j > 0 && A[j-1] > A[j] Hacer // c3*n
      temp <- A[j]; // c4 * ∑_{i=1}^{n} i
      A[j] <- A[j-1]; // c5 * ∑_{i=1}^{n} i
      A[j-1] <- temp; // c6 * ∑_{i=1}^{n} i
      j <- j - 1; // c7 * ∑_{i=1}^{n} i
    FinMientras
  FinPara
FinProceso
```

$$T(n) = c.n^2 + c'.n + c''$$

How do we compute $\sum_{i=1}^{n} i$?

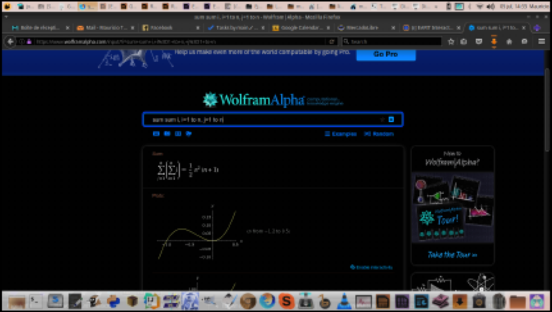
It has been proven that

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

http://www.math.com/tables/expansion/power.htm

How you need help with sums?

Use this tool



https://www.wolframalpha.com/

Insertion sort (2)

```
Proceso InsertionSort
  Para i <- 0 hasta n-1 Hacer // c1*n + c2
    j <- i; // n
    Mientras j > 0 && A[j-1] > A[j] Hacer // c3*n
      temp <- A[j]; // c4 * Σ_{i=1}^{n} i
      A[j] <- A[j-1]; // c5 * Σ_{i=1}^{n} i
      A[j-1] <- temp; // c6 * Σ_{i=1}^{n} i
      j <- j - 1; // c7 * Σ_{i=1}^{n} i
    FinMientras
  FinPara
FinProceso
```

$T(n) = c.n^2 + c'.n + c''$ is $O(n^2)$... why?

Insertion sort: Proof

1. $T(n) = c.n^2 + c'.n + c''$
2. $c.n^2 + c'.n + c''$ is $O(c.n^2 + c.n + c'')$, by Def. of O
3. $O(c.n^2 + c'.n + c'') = O(c.n^2)$, by Rule of Sums
4. $O(c.n^2) = O(n^2)$, by Rule of Products
5. Therefore, $T(n) = c.n^2 + c'.n + c''$ is $O(n^2)$

Insertion sort: Proof

1. $T(n) = c.n^2 + c'.n + c''$
2. $c.n^2 + c'.n + c''$ is $O(c.n^2 + c.n + c'')$, by Def. of O
3. $O(c.n^2 + c'.n + c'') = O(c.n^2)$, by Rule of Sums
4. $O(c.n^2) = O(n^2)$, by Rule of Products
5. Therefore, $T(n) = c.n^2 + c'.n + c''$ is $O(n^2)$

Insertion sort: Proof

1. $T(n) = c.n^2 + c'.n + c''$
2. $c.n^2 + c'.n + c''$ is $O(c.n^2 + c.n + c'')$, by Def. of O
3. $O(c.n^2 + c'.n + c'') = O(c.n^2)$, by Rule of Sums
4. $O(c.n^2) = O(n^2)$, by Rule of Products
5. Therefore, $T(n) = c.n^2 + c'.n + c''$ is $O(n^2)$

Insertion sort: Proof

1. $T(n) = c.n^2 + c'.n + c''$
2. $c.n^2 + c'.n + c''$ is $O(c.n^2 + c.n + c'')$, by Def. of O
3. $O(c.n^2 + c'.n + c'') = O(c.n^2)$, by Rule of Sums
4. $O(c.n^2) = O(n^2)$, by Rule of Products
5. Therefore, $T(n) = c.n^2 + c'.n + c''$ is $O(n^2)$

UNIVERSIDAD EAFIT

Insertion sort: Proof

1. $T(n) = c.n^2 + c'.n + c''$
2. $c.n^2 + c'.n + c''$ is $O(c.n^2 + c.n + c'')$, by Def. of O
3. $O(c.n^2 + c'.n + c'') = O(c.n^2)$, by Rule of Sums
4. $O(c.n^2) = O(n^2)$, by Rule of Products
5. Therefore, $T(n) = c.n^2 + c'.n + c''$ is $O(n^2)$

Quiz questions

- Compute the sum of the elements of an array is $O(n)$
- Compute the maximum element of an array is $O(n)$
- Insertion sort is $O(n^2)$

References

- Please learn how to reference images, trademarks, videos and fragments of code.
- Avoid plagiarism

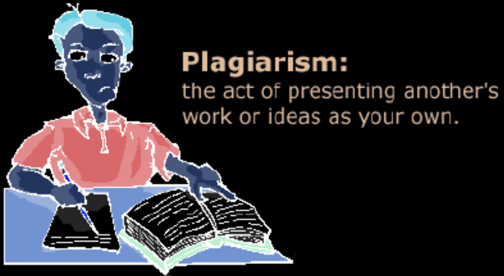


Figure: Figure about plagiarism, University of Malta [Uni09]

University of Malta.

Plagarism — The act of presenting another's work or ideas as your own, 2009.

[Online; accessed 29-November-2013].

Further reading

- Complexity of algorithms
  - Brassard y Bratley, Fundamentos de Algoritmia. Capítulo 3: Notación asintótica. Páginas 98 - 106.